A Formal Framework for Modeling and Analysis of Organizations

Viara Popova and Alexei Sharpanskykh
Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{popova, sharp}@cs.vu.nl

Abstract. This paper introduces a formal framework for modeling and analysis of organizations. It allows representing a great variety of organizational concepts and relations that are structured into a number of dedicated perspectives (or views), similar to the ones defined in GERAM [3]. In contrast to many existing enterprise architectures the proposed framework has formal foundations based on the order-sorted predicate logic. This formal basis enables different types of analysis of organizational specifications both of particular views and across different views. Furthermore, the framework provides support for real time management of organizational processes. The framework has been applied in a number of case studies, one of which is discussed in this paper.

1   Introduction

Nowadays, many organizations employ automated management systems, based on a great variety of enterprise architectures [3] (e.g., CIMOSA, ARIS, Zachman, PERA, GRAI/GIM, TOVE). Based on the analysis of a large number of the existing architectures, the IFIP/IFAC Task Force has developed the Generalized Enterprise Reference Architecture and Methodology (GERAM) [3], which forms a basis for comparison of the existing architectures and serves as a template for the development of new architectures. GERAM identifies the essential characteristics for methods, models and tools required to build and to maintain the integrated enterprise at different phases of its life-cycle. Moreover, to reduce the complexity of enterprise modeling GERAM identifies a number of particular views on enterprises (e.g., function, information, resource, organization) and defines a standard vocabulary of concepts that may be used in the context of these views. The existing architectures conform to the recommendations of GERAM to a variable degree [3]. Although many architectures include a rich ontological basis for creating models of different views, most of them provide only a limited support for automated analysis of these models, addressed in the category *Enterprise Engineering Tools* of GERAM,

primarily due to the lack of formal foundations in these frameworks. Formal analysis is particularly useful for checking the correctness of enterprise models, for inspecting and improving efficiency and effectiveness of the enterprise operation by identifying inconsistencies and performance bottlenecks, as well as for controlling the actual execution of organizational scenarios and evaluating organizational performance. Moreover, analysis methods (e.g. simulation) may be used to investigate and predict organizational behavior and performance under different conditions.

Within several frameworks analysis methods limited to particular views have been developed (e.g., process-oriented modeling techniques for the function view [1, 2, 3, 10], ABC-based techniques for the performance view [26]). However, since different modeling views are related to each other, this should also be reflected in the analysis methods. Analysis performed across different views allows investigating a combined influence of factors from different views on the organizational behavior, thus, the designer is provided with more rigorous and manifold analysis possibilities than by using analysis techniques dedicated to a particular view only. The need for such analysis techniques is identified in [12]. A uniform formal basis (syntax and semantics) underlying different views facilitates the development of cross-view analysis methods. In [12] an integrated framework for process and performance modeling is described that incorporates accounting/business parameters into a formal process modeling approach based on Petri-nets. However, key aspects as authority and power relations, organizational and individual goals, individual behavior are not considered. Another formal framework for business process modeling is described in [14] focusing on the formal goal-oriented modeling using the situation calculus. Modeling and analysis of processes and other organizational concepts are not properly addressed in this framework. A formal framework for verifying models specified in Unified Enterprise Modeling Language (UEML) is proposed in [9]. It identifies a general idea to use conceptual graphs for verifying enterprise models; however, neither technical nor experimental results are provided to support this idea.

Since individuals often exert a significant influence on the organizational dynamics, also aspects related to human behavior should be explicitly considered in enterprise architectures. In particular, by modeling motivational and intentional aspects of humans, an organization can flexibly (re)organize the work of its employees to improve the productivity. The extensive theoretical basis on modeling humans in organizational context developed in social science (e.g., theory of needs [17], expectancy theory [27]) is largely ignored in the existing architectures.

To address the issues and shortcomings identified above, this paper proposes a formal framework for organizational modeling and analysis that:

(1) has a high expressivity to represent static and dynamic aspects of different views on organizations, similar to the ones defined in GERAM;
(2) allows the representation and analysis of organization models at different levels of abstraction in order to handle complexity and increase scalability;
(3) enables formal verification and validation of models of different views;
(4) enables simulation for experimenting and testing hypothesis on the organizational behaviour under different circumstances;
(5) proposes manifold computational analysis methods across multiple views;
(6) incorporates agent-based models of individuals based on social theories;
(7) supports and controls the execution of organizational scenarios and the evaluation of organizational performance.

The framework addresses design, implementation and operation life-cycle phases of GERAM to a greater extent than identification, concept and requirements phases.

The framework proposes a wide spectrum of means for modeling and analysis of structures and dynamics of organizations of different types. In particular, the framework allows modeling mechanistic organizations that represent systems of hierarchically linked job positions with clear responsibilities that operate in a relatively stable (possibly complex) environment. At the same time the framework proposes modeling and analysis means for organic organizations characterized by highly dynamic, constantly changing, organic structure with non-linear behavior. Although the structure and behavioral rules for organic organizations can be hardly identified and formalized, nevertheless by performing agent-based simulations with changing characteristics of proactive agents useful insights into functioning of such organizations can be gained. Furthermore, the framework supports reuse of parts of models constructed within particular organizational views.

The focus of this paper is on the general framework design and analysis methods involving concepts and relation of more than one view, thus, integrating the four views in a coherent and consistent modeling framework. The separate views with their analysis techniques are presented in details elsewhere [6, 18, 19, 20, 21, 23, 24].

The paper is organized as follows. Section 2 describes the formal foundations of the proposed framework. The case study used for the illustration of the framework is introduced in Section 3. Section 4 gives a brief overview of the four modeling views. The issues of design of organization models using the framework are discussed in Section 5. The methods for the organizational analysis using the framework are described in Section 6. Finally, Section 7 concludes the paper.

2 Formal foundations of the proposed framework

In line with GERAM, the proposed framework introduces four interrelated views: performance-oriented, process-oriented, organization-oriented, and agent-oriented. The first-order sorted predicate logic [16] serves as a formal basis for defining dedicated modeling languages for each view. These languages provide high expressivity for conceptualizing a variety of concepts and relations and allow expressing both quantitative and qualitative aspects of different views.

To express temporal relations in specifications of the views, the dedicated languages of the views are embedded into the Temporal Trace Language (TTL) [4, 6, 25], which is a variant of the order-sorted predicate logic. In TTL the organizational dynamics are represented by a trace, i.e. a temporally ordered sequence of states. Each state is characterized by a unique time point and a set of state properties that hold (i.e., are true). State properties are specified using the dedicated language(s) of the view(s). Temporal (or dynamic) properties are defined in TTL as transition relations between state properties. For the description of the formal syntax and semantics, and examples of use of TTL we refer to [25].

Both specifications in the dedicated languages of the views and in TTL are suitable for performing computations. In particular, in [25] it is shown that any TTL formula can be automatically translated into executable format that can be implemented in most commonly used programming languages.

Within every view a set of structural and behavioral *constraints* imposed on the specifications of the view can be identified. Formally, this set is represented by a *logical theory* that consists of formulae constructed in the standard predicate logic way [16] from the terms of the dedicated language of the view (and of TTL if temporal relations are required). Since the views are related to each other by sets of common concepts, also these concepts can be used in the constraints expressions. A specification of the view is *correct* if the corresponding theory is satisfied by this specification, i.e., all sentences in theory are true in the logical structure(s) corresponding to the specification. The constraints are divided in two groups: (1) *generic constraints* need to be satisfied by any specification of the view; (2) *domain-specific constraints* are dictated by the application domain and may be changed by the designer. Two types of generic constraints are considered: (1) *structural integrity and consistency constraints* based on the rules of the specification composition; (2) *constraints imposed by the physical world*. Domain-specific constraints can be imposed by the organization, external parties or the physical world of the specific application domain. The algorithms for the verification of the correctness of specifications of every view w.r.t. different types of constraints have been developed and implemented, and will be discussed in Section 6.

3 Introduction to the case study

The proposed approach was applied for modeling and analysis of an organization from the security domain within the project CIM (Cybernetic Incident Management, see http://www.almende.com/cim/). The main purpose of the organization is to deliver security services to different types of customers. The organization has well-defined multi-level structure that comprises several areas divided into locations with predefined job descriptions for employees (approx. 230.000 persons).

The examples in this paper are related to the planning of assignment of security officers to locations. The planning process consists of forward (long-term) planning and short-term planning. Forward planning is the process of creation, analysis and optimization of forward plans for the allocation of security officers based on custommer contracts. It is performed by forward planners from the forward planning group. During the short-term planning, plans for the allocation of security officers in a certain area for a short term (a week) are created and updated based on a forward plan and up-to-date information about the security officers. Based on short term plans, daily plans are created. Short-term planning is performed by area planning teams.

4 Modeling views

In this section, the views of the proposed framework will be presented. Three of them, process-oriented, performance-oriented and organization-oriented, have prescriptive character and define the desired behavior of the organization. The fourth view, agent-oriented, describes and integrates agents into the framework.

4.1 Process-oriented view

The process-oriented view of the framework contains information about the organizational functions, how they are related, ordered and synchronized and the resources they use and produce. The main concepts are: task, process, resource type and resource which, together with the relations between them, are specified in the formal language L_{PR}. A *task* represents a function performed in the organization and is characterized by *name*, *maximal* and *minimal duration*. Tasks can range from very general to very specific. General tasks can be decomposed into more specific ones using AND- and OR-relations thus forming hierarchies.

A *workflow* is defined by a set of (partially) temporally ordered *processes*. Each process is defined using a task as a template and inherits all characteristics of the task. Decisions are also treated as processes associated with decision variables taking as values the possible decision outcomes. The (partial) order of execution of processes in the workflow is defined by sequencing, branching, cycle and synchronization relations specified by the designer. Part of the workflow describing the short-term planning process in the organization from the case study is given in Fig. 1 seen at two different levels of abstraction.
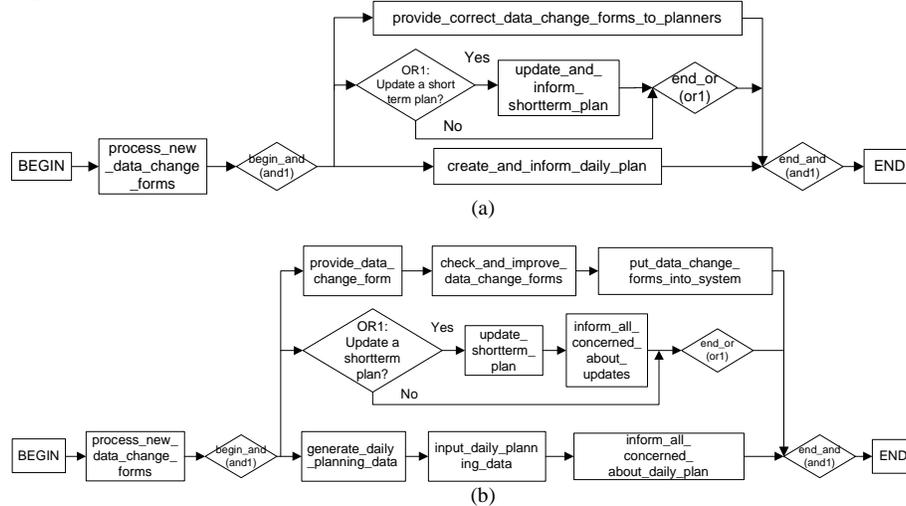

Fig.1 Part of the workflow describing the short-term planning process for the case study

Tasks use/consume/produce resources of different types. Resource types describe tools, supplies, components, data or other material or digital artifacts and are characterized by *name*, *category* (discrete, continuous), *measurement_unit*, *expiration_duration* (the length of the time interval when a resource type can be used). Resources are instances of resource types and inherit their characteristics, having, in addition, *name* and *amount*. Some resources can be shared, or used simultaneously, by a set of processes (e.g., storage facilities, transportation vehicles). Alternative sets of processes sharing a resource can be defined.

Using the language, generic and domain-specific constraints can be defined. Generic constraints include structural constraints on the correctness of the workflow, task and resource hierarchies and constraints from the physical world. An example of

a structural generic constraint is: "For every and-decomposition of a task, the minimal duration of the task is at least the maximal of all minimal durations of its subtasks". An example of a physical world generic constraint is: "For every process that uses certain amount of a resource as input, without consuming it, either at least that amount of resource of this type is available or can be shared with another process at every time point during process execution". Domain-specific constraints can be added by the designer using templates.

L_{PR} has some similarities with and distinctions from other process modeling languages [1, 2, 10]. In particular, it realizes the most commonly used workflow patterns identified in [2] extended with time parameters (e.g., sequence and parallel execution, synchronization, loops). In comparison with other approaches [1, 3], L_{PR} provides a more extensive means for resource modeling (e.g. shared resources). More details on the process-oriented modeling using L_{PR} can be found in [21].

4.2 Performance-oriented view

Central notions in the performance-oriented view are goal and performance indicator (PI). A PI is a quantitative or qualitative indicator that reflects the state/progress of the company, unit or individual. The characteristics of a PI include, among others: *type* – continuous, discrete; *unit* of measurement; *time_frame* – the length of the time interval for which it will be evaluated; *scale* of measurement; *source* – the internal or external source used to extract the PI: company policies, mission statements, business plan, job descriptions, laws, domain knowledge, etc.; *owner* – the performance of which role or agent does it measure/describe; *hardness* – soft or hard, where soft means not directly measurable, qualitative, e.g. customer's satisfaction, company's reputation, employees' motivation, and hard means measurable, quantitative, e.g., number of customers, time to produce a plan. For the case study, 33 PIs were identified examples of which are given below:

PI name: PI5*;*
Definition: average correctness of plans
Type: discrete*; Time_frame:* month*;*
Scale: very_low-low-med-high-very_high*;*
Source: mission statement, job descriptions*;*
Owner: forward/daily planning departments
Hardness: soft*; ...*

PI name: PI27*;*
Definition: time to create new short-term plan
Type: continuous*; Time_frame:* month
Scale: REAL*; Unit:* hour*;*
Source: job descriptions
Owner: daily planning departments
Hardness: hard*;...*

PIs can be related through various relationships. The following are considered in the framework: (strongly) positive/negative causal influence of one PI on another, positive/negative correlation between two PIs, aggregation – two PIs express the same measure at different aggregation levels. Such relationships can be identified using e.g. company documents, domain knowledge, inference from known relations, statistical or data mining techniques, knowledge from other structures of the framework. Using these relations, a graph structure of PIs can be built.

Based on PIs, PI expressions can be defined as mathematical statements over PIs that can be evaluated to a numerical, qualitative or Boolean value. They are used to define goal patterns. The *type* of a goal pattern indicates the way its property is checked: *achieved (ceased)* – true (false) for a specific time point; *maintained (avoided)* – true (false) for a given time interval; *optimized* – if the value of the PI expression has increased, decreased or approached a target value for a given interval.

Goals are objectives that describe a desired state or development and are defined by adding to goal patterns information such as desirability and priority. The characteristics of a goal include, among others: *priority*; *evaluation type* –

achievement goal (based on achieved/ceased pattern – evaluated for a time point) or development goal (based on maintained/avoided/optimized pattern – evaluated for a time interval); *horizon* – for which time point/interval should the goal be satisfied; *hardness* – hard (satisfaction can be established) or soft (satisfaction cannot be clearly established, instead degrees of *satisficing* are defined); *negotiability*. Examples of goals identified for the case study are given below:

Goal name: G3.2	*Goal name:* G3.1.1.1
Definition: It is required to maintain high efficiency of allocation of security officers	*Definition:* It is required to achieve that the time to update a short-term plan given operational data is at most 48 hours
Priority: high*; Horizon:* long-term	
Evaluation type: development goal	*Priority:* high*; Horizon:* short-term
Perspective: management, customer	*Evaluation type:* achievement goal
Hardness: soft*; Negotiability:* negotiable*, ...*	*Perspective:* management
	Hardness: hard*, Negotiability:* negotiable*,...*

A goal can be refined into sub-goals forming a hierarchy. Information about the satisfaction of lower-level goals can be propagated to determine the satisfaction of high-level goals. A goal can be refined into one or more alternative goal lists of AND-type or balanced-type (more fine-tuned ways of decomposition - inspired by the weighted average function) [19]. For each type, propagation rules are defined. Fig. 2 shows a part of the goals hierarchy built for the case study.

Using the concepts and relations of the performance-oriented view, constraints can be formulated. An example of a generic structural constraint is: "If two PIs are related by an aggregation relation then they should have the same type and measurement unit."

Modeling goals is supported to a various degree by a number of existing frameworks in enterprise modeling; however the concept of a PI has been largely ignored. Our approach [19, 20] differs in explicitly representing PIs and the link between a goal and the PI that will measure its satisfaction. Besides the relationships between PIs can be represented and used for reasoning at the design phase.
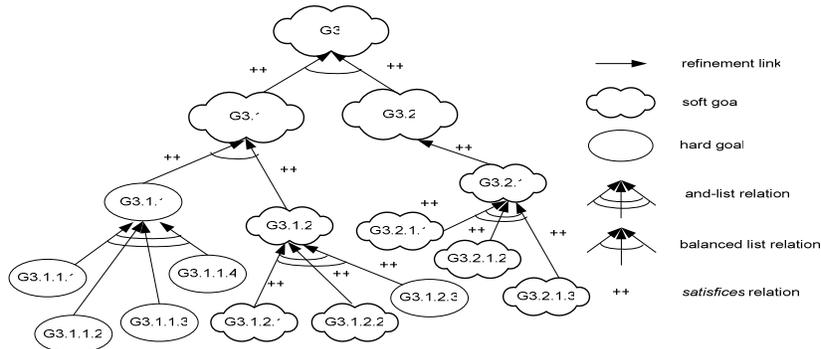

Fig. 2 A part of the goal hierarchy for the case study

4.3   Organization-oriented view

In the organization-oriented view organizations are modeled as composite roles that can be refined iteratively into a number of (interacting) composite or simple roles, representing as many aggregation levels as needed. The refined role structures correspond to different types of organization constructs (e.g., groups, units, departments). Yet many of the existing modeling frameworks are able to represent only two or three levels of abstraction: the level of a role, the level of a group composed of roles, and the overall organization level, as in [13]. The organization-

oriented view provides means to structure and organize roles by defining interaction and power relations on them. First, interaction relations are discussed.

One of the aims of an organizational structure is to facilitate the interaction between the roles that are involved into the execution of the same or related task(s). Therefore, patterns of role interactions are usually reflected in an organization structure. Each role has an input and an output interface, which facilitate in the interaction (in particular, communication) with other roles and the environment. Role interfaces are described in terms of interaction (input and output) ontologies: a signature specified in order-sorted logic. Generally speaking, an input ontology determines what types of information are allowed to be transferred to the input of a role (or of the environment), and an output ontology predefines what kinds of information can be generated at the output of a role (or of the environment). In particular, to specify a special type of interaction – a speech act s_act (e.g., inform, request, ask) with the content message the ontologies of both role-source r1 and role-destination r2 should include the predicate communicate_from_to(r1:ROLE, r2:ROLE, s_act:SPEECH_ACT, message:STRING). Roles that are allowed to interact are connected by an interaction link that indicates the direction of the interaction (see Fig. 3).

The representation of the environment may vary in different organizational specifications. In particular, in some cases it can be defined by a set of objects with certain properties and states and by causal relations between objects. While in other cases the dynamics of the environment is described by (high-level) processes and trends (e.g. changes of the market situation, natural environmental oscillations).

Since roles may have composite structure, interaction processes can be modeled at different levels of abstraction. Interaction relations between roles can also be depicted in a modular way; thus, scalability of graphical representation is achieved. Moreover, interaction relations specified at the generalized level, represent templates that can be instantiated for a particular case. An instantiated model is obtained from a template by unfolding generic relations between roles and by creating new role instances. For example, the documents of the organization from the case study define standard patterns of interaction between the forward planner and the daily planner roles that can be modeled at the generalized (template) level. However, for a more detailed analysis of the organizational dynamics, a more specific representation defining interaction relations between particular role instances of the forward planner and the daily planner roles (e.g., from different planning teams) is needed (see Fig. 3). For a more detailed description of the modeling of interaction relations at different levels of abstraction and generalization we refer to [6].

Besides interaction relations, also power relations on roles constitute a part of the formal organizational structure. Formal organizational power (authority) establishes and regulates normative superior-subordinate relationships between roles. Authority relations are defined w.r.t. tasks. In the context of the running example the relation is_subordinate_of_for(Daily_PlannerA, Team_Leader1, daily_planning) means that role Daily_PlannerA is a subordinate of role Team_Leader1 w.r.t. the task daily_planning.

Roles have rights and responsibilities related to different aspects of tasks (e.g., execution, monitoring, consulting, and making technological and/or managerial decisions). For example, is_responsible_for(Daily_PlannerB, execution, inform_about_daily_plan) expresses execution responsibility of role Daily_PlannerB for task inform_about_daily_plan.

A number of generic constraints have been identified in this view. For example, "to assign responsibility for some aspect of a task, a role should have the responsibility to make managerial decisions and be the superior of a role, to which the responsibility is assigned".
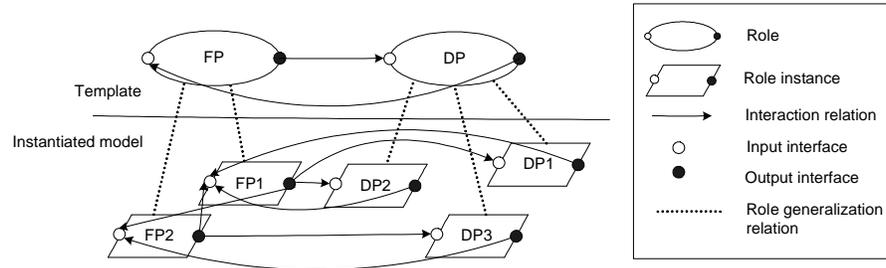
Fig. 3 The graphical representation of interaction relations between the roles Forward Planner (FP) and Daily Planner (DP) (the template) and their instances (the instantiated model)

Roles with managerial rights may under certain conditions authorize and/or make other roles responsible for certain aspects of task execution. In many modern organizations rewards and sanctions form a part of authority relation, thus, they are explicitly defined by appropriate language constructs. Specific conditions (e.g., temporal, situational) under which authority relations may be created/maintained/dissolved are defined by executable rules expressed by TTL formulae. For more details on specifying authority relations in organizations of different types see [23].

4.4 Agent-oriented view

To create realistic organization models, in addition to formal (explicitly identified, documented) aspects, also informal aspects of human behavior in the organizational context should be considered. The computational organization theory [7] has a long tradition of modeling human organizations using the agent paradigm, also used in the proposed framework. Models of agents defined in the agent-oriented view are based on psychological and social theories receiving the most empirical support [17, 27].

An agent is defined as an autonomous entity able to interact (e.g., by observations and actions) with other agents and the environment. Agents are characterized by a set of capabilities (i.e., knowledge and skills) and personal traits. Knowledge of an agent comprises facts and procedures, of which the agent has confident understanding. Skills describe developed abilities of agents to use effectively and readily their knowledge for the performance of tasks. In the literature four types of skills relevant in the organizational context are distinguished: technical, interpersonal, problem-solving/decision-making and managerial skills. Every skill of an agent is associated with a performance indicator. Furthermore, for each skill a numerical value of the skill development that changes over time is defined.

Personal traits are divided into five broad categories discovered in psychology [22]: openness to experience, conscientiousness, extroversion, agreeableness, and neuroticism. Using sets of capabilities and traits, several characteristic types of agents (e.g. "self-confident professional", "intrinsically motivated novice", "submissive employee") are defined that are relevant in different organizational settings.

An agent can be allocated to an organizational role if s/he possesses the necessary capabilities and traits defined as requirements for the role. In the case study, the role Daily_Planner requires the agent to have knowledge and technical skills related to daily planning, as well as some interpersonal skills. The company also defined requirements on personal traits related to conscientiousness (self-discipline, responsibility, aim for achievement) and agreeableness (cooperative work style).

To model the dynamics of an agent situated in the organizational context, the agent's intentional and motivational aspects are considered in the agent-oriented view. Each agent has a set of needs that s/he strives to satisfy. At present, a widely accepted categorization of needs in social science is: (1) extrinsic needs associated with biological comfort and material rewards; (2) social interaction needs - the desire for social approval, affiliation and companionship; (3) intrinsic needs that concern the desire for self-development, self-actualization, mastery and challenge. The level of satisfaction and importance of different types of individual needs change with time causing change in priorities of individual goals related to these needs. The highest motivation is demonstrated by an agent w.r.t. actions (e.g., the execution of organizational tasks) that (significantly) contribute to the satisfaction of his/her primary goals. An organization that recognizes primary goals of its agents often can arrange their work and provide incentives so that the agents are constantly stimulated to adopt the behavior that also ensures the satisfaction of organizational goals.

For reasoning about the agent motivation and work behavior, Vroom's version of the expectancy theory [27] is used which establishes causal dependencies between a number of individual, organizational and environmental parameters and the agent's motivation to perform certain action (e.g., process). The expectancy theory is one of the few organization theories that can be made operational and used for simulation.

5   Design issues

The general approaches to organization design differ w.r.t. the presence and involvement of the concerned agents. The design can be performed without having in mind specific agents - the necessary agent profiles are composed at the later design stages based on the considered/designed tasks. Organizational design can also be performed w.r.t. a (partially) known set of agents who will take roles in the organization. Thus agents' skills and traits can be taken into account. Sometimes the agents are not only known but they have some degree of power to steer the design process.

The design process often starts with the identification of one or more high-level goals which play the role of the driving force behind the design process. These goals (initially still informally defined) should answer the question: why should the organization exist and what purpose will it serve? Such goals can be identified by the designer or emerge through communication and/or negotiation between the involved agents. In the second case the resulting organizational goals reflect to some extent the individual goals of the participating agents. In this way some possible future conflicts between individual and organizational goals are prevented early. If conflicts do appear, they can be dealt with through negotiation and redesign at the later stages.

The higher-level goals are often more abstract and, through refinement, more specific, easier to evaluate, goals are formulated. Also, often the higher-level goals are long-term, strategic goals while their sub-goals are shorter-term tactical or operational goals. The leaves of the hierarchies should be goals formulated so that the corresponding PIs can clearly be associated to the processes in the workflow. In this way the satisfaction of every goal in the hierarchies can be evaluated.

Also at the earlier stage of the design process one or more general tasks are identified giving an answer to the question: what should the organization do? For identifying these tasks sometimes only the defined goals are considered. However

when the involved agents are (partially) known, the definition of tasks can be based on the available skills and experience as well. These tasks are later refined to task hierarchies. For the tasks, the used / produced resource types are identified which can also form hierarchies. Based on the tasks, processes are defined and organized into workflows that can represent different levels of abstraction. The level of elaboration of these structures can depend on the type of the organization. In mechanistic organizations [22] the procedures are prescribed to a great degree of detail which should result in more elaborate structures refined to simple tasks and processes. In organic organizations (e.g., adhocracies) the procedures are described at a higher level of abstraction leaving more freedom to the agents to choose how to perform them which should result in less deep task hierarchies and less elaborate workflows.

The design process can follow different paths through the views and concepts but several general guidelines can be formulated. When an informally defined goal is being formalized and made more precise this should be reflected on the PI structure - often this means that a new PI is defined or an existing one is revised. A change in the goal hierarchy should also be reflected on the task hierarchy by identifying new or existing tasks that can realize the new or revised goals. A change in the task hierarchy often brings changes to the current workflow design. Adding or revising processes in the workflow might give rise to new PIs that need to be monitored. When a PI is proposed it should be decided on its level of importance in order to find out if a new goal should be formulated based on it. The definition of roles is based on the currently defined tasks and processes. Fig.4 shows the main dependencies between concepts and structures in the framework which guide the design process.
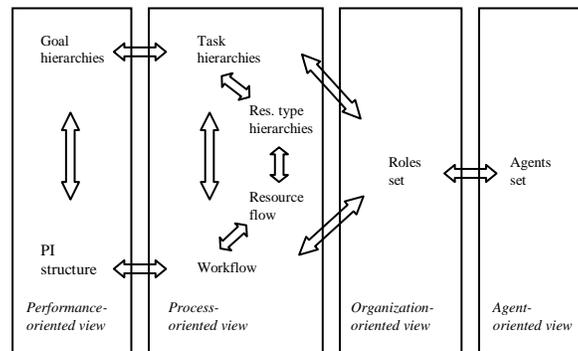

Fig. 4 Dependencies between the structures of the four views

Power and authority relations between the defined roles are usually assigned at the later stages of the design process. However different general schemes can be predefined and committed to by the designer at the earlier stages as well leaving the details for later. Such schemes reflect different types of organizations identified in organization theory such as: hierarchical, flat or team-based organizations which differ in the way the power is distributed, granted or accepted by the roles (agents).

The choice of scheme should be driven by an analysis of the environment in which the organization should operate. For example a relatively stable environment tolerates a well-defined hierarchical structure which can help the organization to operate more efficiently. A changing environment can be addressed by designing a lighter, more flexible and dynamic structure that can easily adapt to the changes. Obviously the environment in which the organization will be situated plays an

important role not only in defining power and authority relations. It needs to be taken into account at every step of the design process and in every view of the framework.

Sometimes instead of designing an organization from scratch, a specification of an existing one needs to be created. Here a wide range of internal or external documents are used, e.g., company policies, job descriptions, mission statement, business plans, procedure descriptions, laws. However even the richest documentation leaves some information unspecified thus it is essential to involve domain experts and managers. In organizational redesign, the issue of maintaining the consistency and correspondence between the structures of different views becomes more complex and the tools for automatic analysis become indispensable.

The framework allows reuse in a number of ways. Libraries of commonly appearing parts of structures (goals and tasks hierarchies, PI-structures, workflow graphs, etc.) can be stored and reused for organizations in the same domain. The research in identifying and classifying important PIs for different domains [e.g. 8, 15] can easily be applied here. Reuse can also be supported by predefined templates for various aspects of different types of organizations (mechanistic, organic, etc.). For example templates for domain-specific constraints can be provided for each view to be customized by the designer. The used tool allows defining parameterized templates (macros) for TTL formulae that can be instantiated in different ways which can also be used as support for designers not skilled in logics. For more details see [21].

6   Analysis methods

The formal foundations of the proposed framework enable three types of automated analysis. The first type focuses on the verification of specifications of every view (i.e., establishing the correctness w.r.t. a set of constraints). The second type addresses the validation of (combined) correct specifications of different views by simulation. Finally, the third type focuses on the analysis of actual executions of organizational scenarios based on (combined) specifications from different views. The three types of analysis are discussed in this order in the rest of this Section.

The verification of the consistency of a PI structure is performed by checking constraints based on the inference rules described in [20]. The inference rules allow generating all correct causality relations between PIs that should hold in the PI structure. Since goal and PI structures are closely related, it is important to guarantee consistency and correspondence of these structures to each other. For this a dedicated consistency check can be performed, based on the constraints described in [19]. For organizations that do not allow conflicts between goals, a number of dedicated techniques for the identification and the resolution of conflicts are proposed in [19].

In the process-oriented view constraints are defined for the three types of structures: workflow, task and resource hierarchies [21] that should be satisfied by specifications. The verification of the correctness of a specification is performed during or at the end of the design process, depending on the type of constraint. Some domain-specific constraints might not (yet) be satisfied for incomplete specifications. The designer can choose when they should be checked. The syntactical check of a specification and the verification of generic constraints are performed at each design step.

Note that workflow specifications can be represented and analyzed at different levels of abstraction. In general, the verification of higher-level specifications is

computationally cheaper than that of more detailed lower-level specifications. Furthermore, a correct high level workflow specification can be refined to a lower level by using the correct hierarchy of tasks, on which the processes of the workflow are based. In such a case the correctness verification of the obtained workflow is guaranteed without additional verification. The verification of interaction relations in composite (multi-level, hierarchical) organizational structures is addressed in [6].

The algorithms developed for the verification of constraints of different types in the proposed framework are more efficient than general-purpose methods for verifying specifications (e.g., model checking [11]).

As shown in [18], correct organizational specifications can be used to guide and to control the actual execution of processes in organizations. The execution data recorded by an enterprise information system and structured in the form of a trace can be checked for conformity to a formal organization (i.e., specifications and constraints defined in particular views). To this end, the relations and constraints specified for particular views are translated into properties expressed in the execution language used for the formalization of the trace [18]. They are checked in real time on the trace. Depending on the type of event that (should) occur(s) in the trace at a certain time point, only a subset of relevant properties is checked at this time point. Moreover, the designer may specify additional properties to be checked in real time.

A trace can also be analyzed after the execution of an organizational scenario is completed. For this type of analysis, next to the properties obtained from the formal organization, the designer may specify in TTL and check other properties. The traces are used to evaluate the PIs associated with the executed processes. These PIs are related to the leaves of the goals hierarchy, thus the satisfaction of these goals can be evaluated. The satisfaction values are propagated upwards to establish the satisfaction of higher-level goals determining the overall organizational performance [18].

Based on correct (combined) specifications of the views, simulation can be performed, in which different types of agents, defined using the concepts from the agent-oriented view, are allocated to the organizational roles. By considering different simulation scenarios of organizational behavior, the validation of organizational specifications can be performed (i.e., checking if the model behaves as expected, corresponds to reality) using the dedicated tool [4, 5].

In the context of the case study the behavior of different types of planners under different organizational and environmental conditions was investigated [24]. The simulation results in Fig.5 are related to a planner agent with initially lacking skills but good learning abilities to improve through processes execution. In the simulation comparable amounts of simple and complex tasks and equal arrival rates of tasks are.used. Fig.5a shows the change of the satisfaction level of the intrinsic needs of the agent, performing tasks under the control of a team leader. The simulation results show that the more experience the agent gains, the less s/he appreciates the leader's involvement. Fig.5b shows a growth of the agent satisfaction when the leader exercises direct control only if the agent lacks experience. The simulation results conform to the empirical evidence [27], which supports the specification's validity.

The simulation tool also provides the possibility to generate the simulation results in the form of a trace. Traces can be used for the validation of specifications by checking dynamic properties in the environment TTL Checker [4]. Such properties should be specified in TTL and may be expressed using the concepts and the relations defined in different views. A detailed explanation can be found in [4, 6].

Simulation based on a correct and valid specification can also be used for predictions on the organization's behavior in different environmental conditions and with different agents as well as for investigating theories from the social sciences.
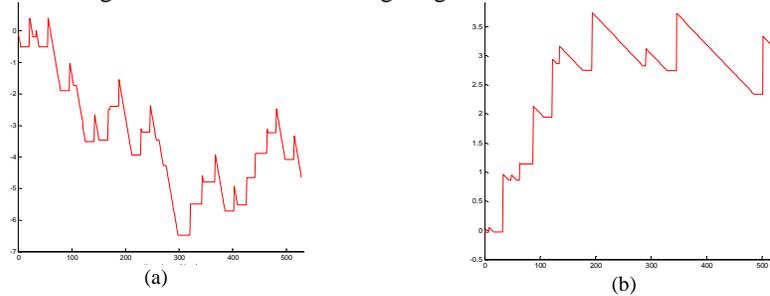

Fig. 5 Change of the satisfaction level of the agent's intrinsic needs (the vertical axis) over time (the horizontal axis) in the case of constant supervision (a) and temporary supervision (b)

7   Conclusions

This paper describes a formal framework for modeling and analysis of organizations. The framework has a rich ontological basis that comprises concepts and relations partitioned into a number of dedicated views similar to the ones defined in GERAM. The introduced modeling framework allows representing different types of organizations ranging from mechanistic to organic. In contrast to many existing architectures, the proposed framework allows performing different types of automated analysis of organizational models (e.g. by verification, validation and simulation) both of particular views and across different views. Moreover, the framework incorporates agent-based models of individuals based on social theories. Organizational models of different views can be represented and analysed at different abstraction levels, which allows handling high complexity and increases scalability of modeling. Finally, the framework allows model reuse that accelerates and facilitates the modeling process.

The views of the proposed framework are formalized based on intuitive, close to the natural, predicate languages, with concepts and relations that can be represented graphically. Currently, the graphical interface is provided for the performance-oriented view, whereas other views are specified textually using the dedicated tools. In the future, modeling related to other views will be also supported graphically.

The application of the proposed framework has been illustrated by an example of an organization from the security domain. The framework was also applied in the context of a case study in logistics (*http://www.almende.com/deal/*). Currently, the framework is used for modeling and analysis of an air traffic control organization.

References

1. W. van der Aalst and K.M van Hee, *Workflow Management: Models, Methods, and Systems* (MIT press, Cambridge, MA, 2002).
2. W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A.P. Barros, Workflow patterns, *Distributed and Parallel Databases* 14(3), 5–51 (2003).

3. P. Bernus, et al. (eds.): *Handbook on Architectures of Information Systems*, Springer-Verlag, Heidelberg (1998) 209-241.

4. T. Bosse, C.M. Jonker, L. van der Meij, A. Sharpanskykh, and J. Treur, Specification and Verification of Dynamics in Cognitive Agent Models. In: *Proceedings of the Sixth Int. Conf. on Intelligent Agent Technology, IAT'06* (IEEE Computer Society Press, 2006), pp. 247-254.

5. T. Bosse, C.M. Jonker, L. van der Meij, and J. Treur, LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn. In: *Proc. MATES'05*. LNAI 3550, edited by T. Eymann et al. (Springer Verlag, 2005) pp. 165-178.

6. E. Broek, C. Jonker, A. Sharpanskykh, J. Treur, and P. Yolum, Formal Modeling and Analysis of Organizations. In *Coordination, Organization, Institutions and Norms in Agent Systems I*, LNAI 3913, (Springer, 2006).

7. K.M. Carley: A comparison of artificial and human organizations. *Journal of Economic Behavior & Organization*, 31(2) 175-191 (1996).

8. F.T.S. Chan, Performance measurement in a supply chain, *International Journal of Advanced Manufacturing Technology* 21(7), 534-548 (2003).

9. V. Chapurlat, B. Kamsu-Foguem, and F. Prunet, A formal verification framework and associated tools for enterprise modeling: Application to UEML, *Computers in industry*, 57, 153-166 (2006).

10. Y.-H. Chen-Burger, A. Tate, and D. Robertson, Enterprise Modelling: A Declarative Approach for FBPML, *European Conference of Artificial Intelligence, Knowledge Management and Organisational Memories Workshop*, 2002.

11. E.M. Clarke, O. Grumberg, and D.A. Peled, *Model Checking* (MIT Press, 2000).

12. N. Dalal, M. Kamath, W. Kolarik, and E. Sivaraman, Toward an integrated framework for modeling enterprise processes, *Communications of the ACM*, 47(3), 83-87 (2004).

13. J. Ferber and O. Gutknecht, A meta-model for the analysis and design of organizations in multi-agent systems. In: *Proceedings of Third International Conference on Multi-Agent Systems (ICMAS'98)* (IEEE Computer Society 1998), pp.128-135.

14. M. Koubarakis and D. Plexousakis. A formal framework for business process modeling and design. *Information Systems*, 27(5), 299–319 (2002).

15. E. Krauth, H. Moonen, V. Popova, and M. Schut, Performance Measurement and Control in Logistics Service Providing, in: *Proceedings of Seventh International Conference on Enterprise Information Systems, ICEIS 2005*, edited by C.-S. Chen et al., pp. 239-247 (2005).

16. M. Manzano, *Extensions of First Order Logic* (Cambridge University Press, 1996).

17. A. H. Maslow, *Motivation and Personality*, 2nd. ed. (New York, Harper & Row, 1970).

18. V. Popova and A. Sharpanskykh, Formal analysis of executions of organizational scenarios based on process-oriented models. To appear in: *Proc. of 21st European Conference on Modeling and Simulation (ECMS'07)*, 2007.

19. V. Popova and A. Sharpanskykh, Formal Modelling of Goals in Agent Organizations. In *Proc. of the AOMS Workshop* (joint with IJCAI 2007), 74-86.

20. V. Popova and A. Sharpanskykh, Modelling Organizational Performance Indicators. In: *Proc. of IMSM'07 conference,* edited by Barros, F. et al., 165–170, (2007).

21. V. Popova and A. Sharpanskykh, Process-Oriented Organization Modeling and Analysis Based on Constraints. Technical Report 062911AI, VUA, http://hdl.handle.net/1871/10545

22. W.R. Scott, *Institutions and organizations* (SAGE Publications, Thousand Oaks 2001).

23. A. Sharpanskykh, Authority and its Implementation in Enterprise Information Systems, Technical Report 070202AI, VUA.

24. A. Sharpanskykh, Modeling of Agent Behavior in the Organizational Context, Technical Report 070323AI, VUA.

25. A. Sharpanskykh and J. Treur, Verifying Interlevel Relations within Multi-Agent Systems. In: *Proc. of the 17th European Conf. on AI, ECAI'06* (IOS Press, 2006), pp. 290-294.

26. K.D. Tham, *Representation and Reasoning About Costs Using Enterprise Models and ABC*, PhD Dissertation, Enterprise Integration Laboratory, Department of Mechanical and Industrial Engineering, University of Toronto, 1999.

27. V.H. Vroom, *Work and motivation* (Wiley, New York, 1964).